CS1010 Programming Methodology

http://www.comp.nus.edu.sg/~cs1010/

UNIT 3

Algorithmic Problem Solving

Leading The World With Asia's Best

Unit 2: Algorithmic Problem Solving

1. Plinko by Psudocode
2. Algorithm
3. Control Structures
4. Examples of Pseudocodes
5. Euclid’s Algorithm

IS PROGRAMING SCARY?

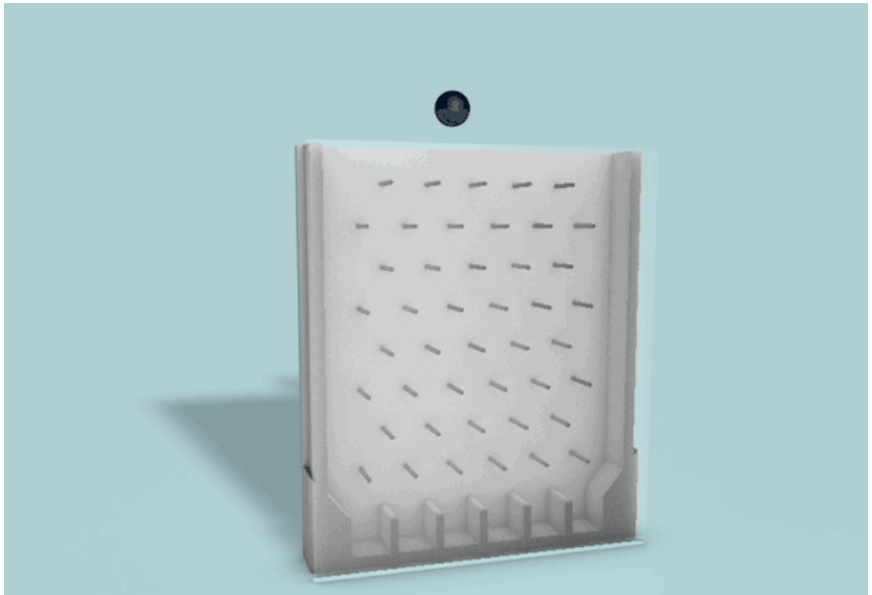
An example of a “program” by pseudocode

What is programming?

Imagine

Running a Booth

- Dropping a ball from the top and you will get
  - 3: Big prize
  - 2: Medium prize
  - 1: Small prize

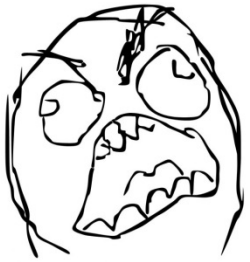


Let's see what we have for prizes

- Giant Teddy Bears
  - X 10

- Water guns
  - X 50

- Candy
  - X 200



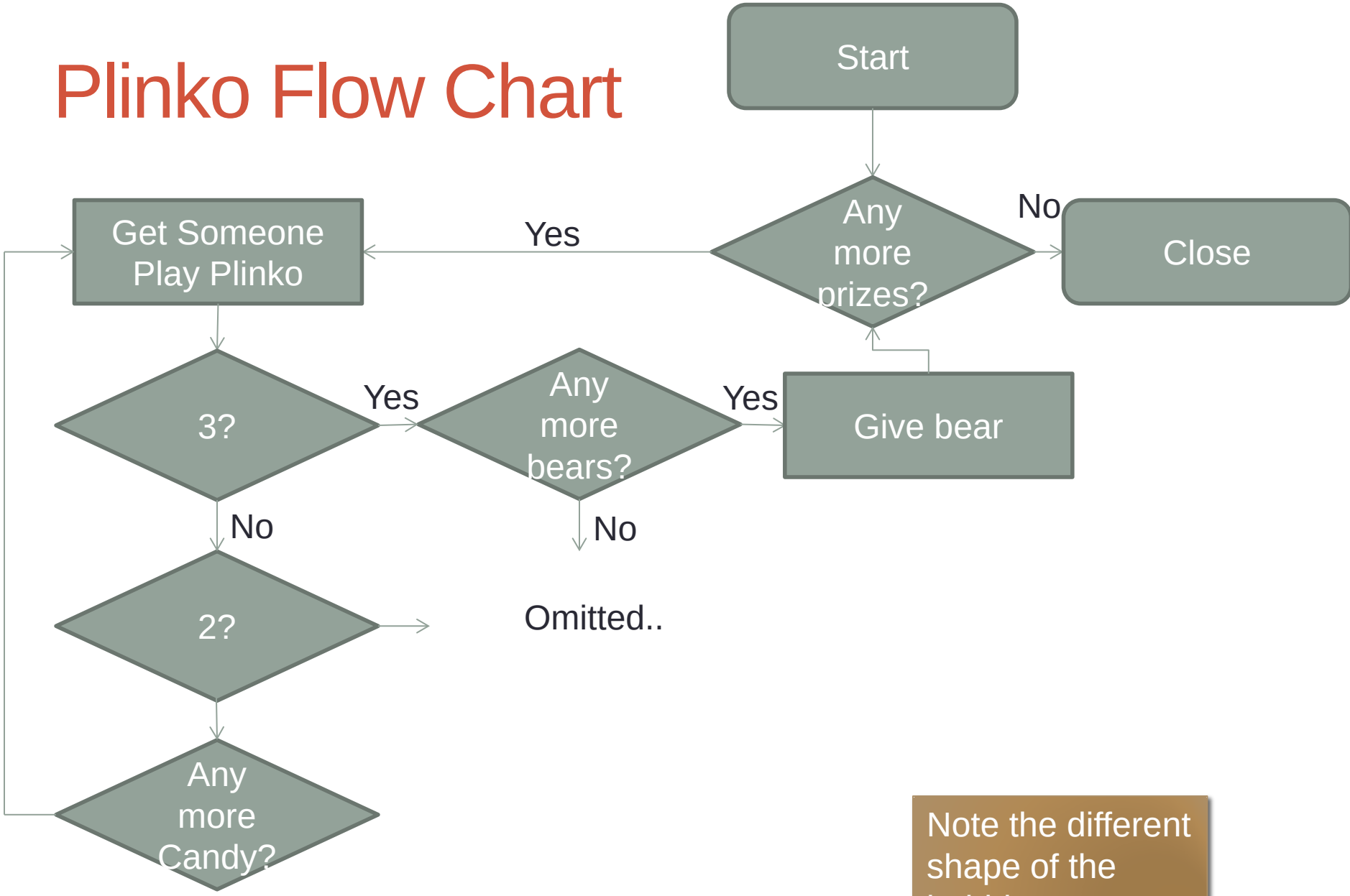
Your Job is

- Run the booth until the end of the day or all prizes given out
  - 3: Giant Teddy Bear
  - 2: Water Gun
  - 1: Candy
- What is the potential problem?
- You may run out of Teddy Bears!
  - Or Water guns or candies
- Any suggestion?

Before Real Coding

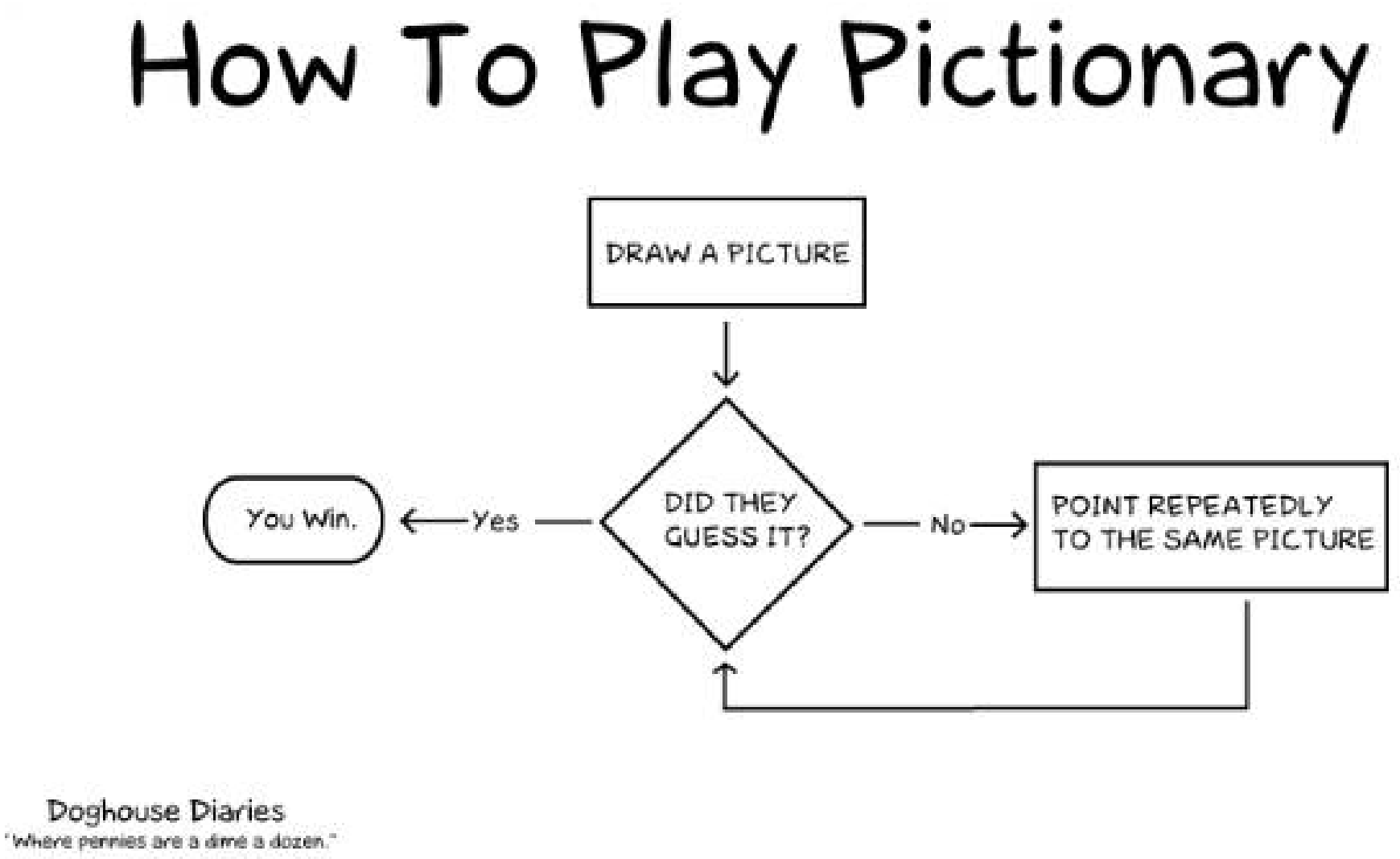
- A Program can be expressed by
  - **Pseudocode**
    - An artificial and informal language that helps programmers develop algorithms.
    - "text-based“
  - Or A **Flow Chart**

Plinko Flow Chart



Note the different shape of the bubbles

Control Structures on Pictionary



Choose Your Own Adventure

From Wikipedia, the free encyclopedia

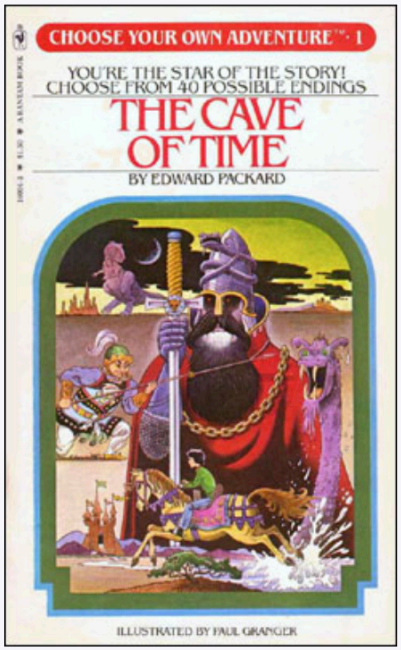
This article is about the trademarked book series. For the genre, see Gamebook. For the TV series, see Lawrence Leung's Choose Your Own Adventure.

Choose Your Own Adventure is a series of children's gamebooks where each story is written from a second-person point of view, with the reader assuming the role of the protagonist and making choices that determine the main character's actions and the plot's outcome. The series was based upon a concept created by Edward Packard and originally published by Constance Cappel's and R. A. Montgomery's Vermont Crossroads Press as the "Adventures of You" series, starting with Packard's *Sugarcane Island* in 1976.[1]

Choose Your Own Adventure, as published by Bantam Books, was one of the most popular children's series during the 1980s and 1990s, selling more than 250 million copies between 1979 and 1998.[2] When Bantam, now owned by Random House, allowed the *Choose Your Own Adventure* trademark to lapse, the series was relaunched by Chooseco, which now owns the trademark. Chooseco does not reissue titles by Packard, who has started his own imprint, U-Ventures.[3]

Contents [hide]

Choose Your Own Adventure



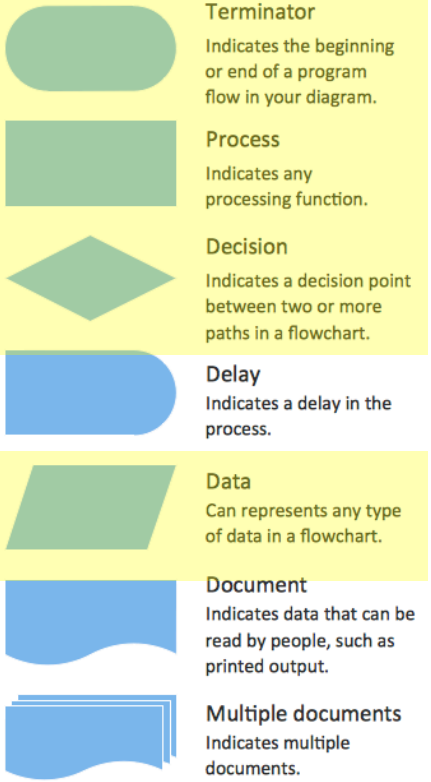
The Cave of Time by Edward Packard, the first book in the series

Cover artist	Paul Granger
Language	English

Format [edit]

Originally created for 7- to 14-year-olds, the books are written in the second person. The protagonist—that is, the reader—takes on a role relevant to the adventure; for example, private investigator, mountain climber, race car driver, doctor, or spy. Stories are generally gender and race neutral, though in some cases, particularly in illustrations, presumption of a male

Flow Chart Bubble Types



Terminator
Indicates the beginning or end of a program flow in your diagram.

Process
Indicates any processing function.

Decision
Indicates a decision point between two or more paths in a flowchart.

Delay
Indicates a delay in the process.

Data
Can represents any type of data in a flowchart.

Document
Indicates data that can be read by people, such as printed output.

Multiple documents
Indicates multiple documents.

Subroutine
Indicates a predefined (named) process, such as a subroutine or a module.

Preparation
Indicates a modification to a process, such as setting a switch or initializing a routine.

Display
Indicates data that is displayed for people to read, such as data on a monitor or projector screen.

Manual input
Indicates any operation that is performed manually (by a person).

Manual loop
Indicates a sequence of commands that will continue to repeat until stopped manually.

Loop limit
Indicates the start of a loop. Flip the shape vertically to indicate the end of a loop.

Stored data
Indicates any type of stored data.

Connector
Indicates an inspection point.

Off-page connector
Use this shape to create a cross-reference and hyperlink from a process on one page to a process on another page.

Off-page connector

Off-page connector

Off-page connector

Or
Logical OR

Summing junction
Logical AND

Collate
Indicates a step that organizes data into a standard format.

Sort
Indicates a step that organizes items list sequentially.

Merge
Indicates a step that combines multiple sets into one.

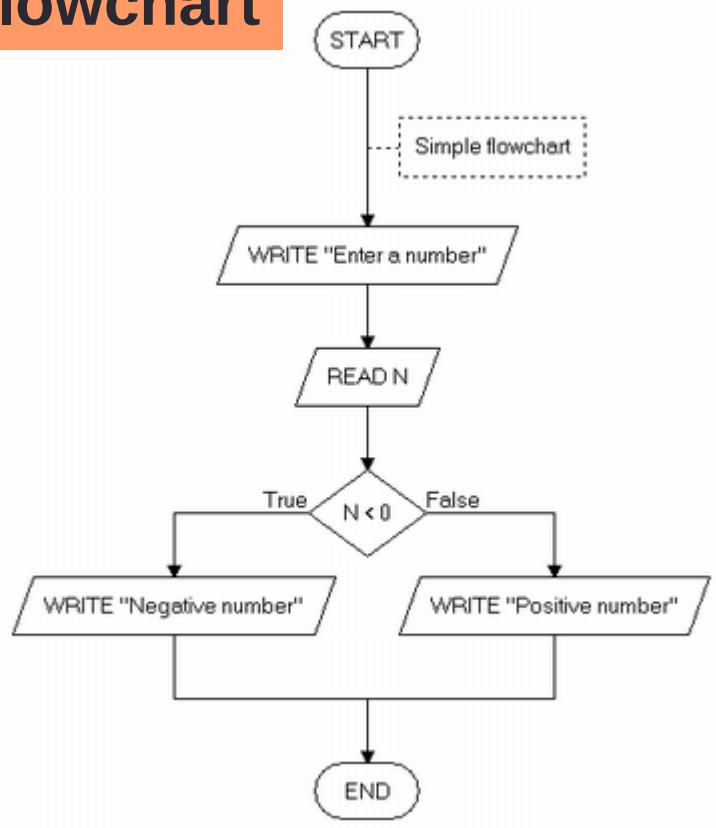
Database
Indicates a list of information with a standard structure that allows for searching and sorting.

Internal storage
Indicates an internal storage device.

Algorithm

- Ways of representing an algorithm:

Flowchart



Pseudocode

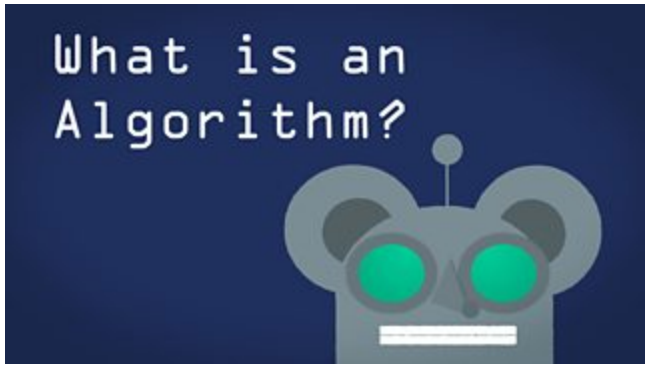
PSEUDOCODE

```
set total to zero

get list of numbers

loop through each number in the list
   add each number to total
end loop

if number more than zero
   print "it's positive" message
else
   print "it's zero or less" message
end if
```

lynda.com

Algorithms

- Named for al-Khwārizmī (780-850)
  - Persian mathematician
- Many ancient algorithms
  - Multiplication: Rhind Papyrus
    - Babylon and Egypt: ~1800BC
  - Euclidean Algorithm: Elements
    - Greece: ~300BC
  - Sieve of Eratosthenes
    - Greece: ~200BC



Algorithm *(noun.)*

Word used by programmers when...
they do not want to explain what they did.

Algorithm (1/3)

- An **algorithm** is a well-defined computational procedure consisting of *a set of instructions*, that takes some value or set of values as *input*, and produces some value or set of values as *output.*

'Algorithm' stems from 'Algoritmi', the Latin form of al-Khwārizmī, a Persian mathematician, astronomer and geographer.
Source: http://en.wikipedia.org/wiki/Algorithm

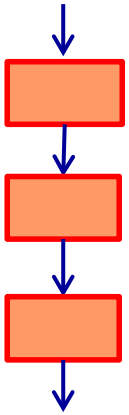
Control Structures

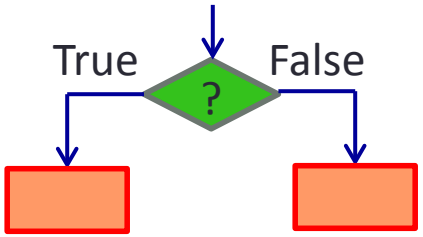
- An algorithm is a set of instructions, which are followed sequentially by default.
- However, sometimes we need to change the default sequential flow.
- We study 3 control structures.

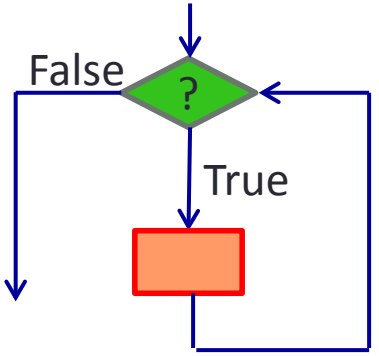
CONTROL STRUCTURES

Control Structures

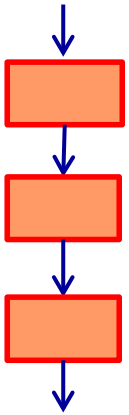
Sequence	• Default
Selection	• Also called branching
Repetition	• Also called loop







Control Structure: Sequence

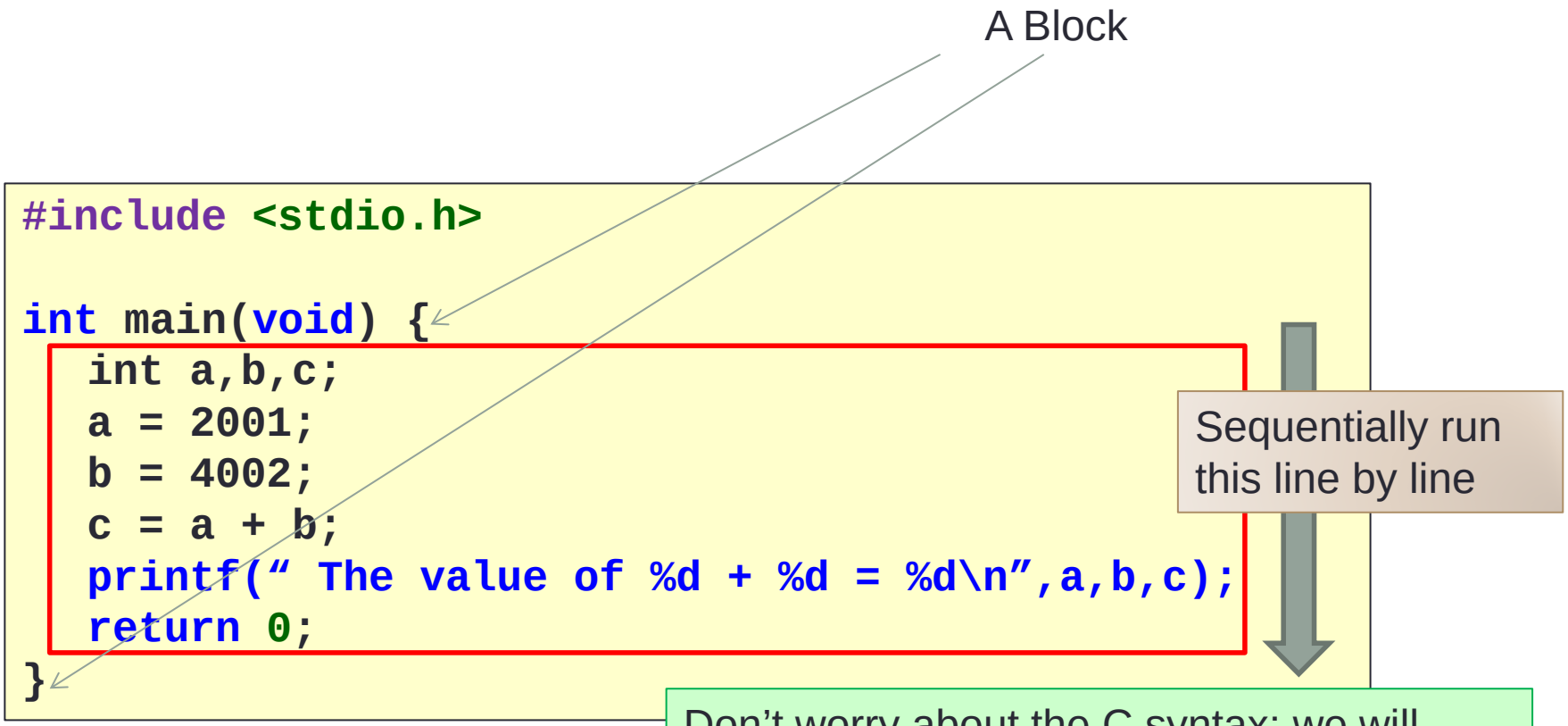


Method 1 Making Vanilla Pound Cake

1. Gather your ingredients. Pound cake is one of the simplest cakes to bake. ...
2. Preheat the oven to 325 degrees.
3. Grease a cake pan. ...
4. Cream the butter and sugar. ...
5. Add the eggs and vanilla. ...
6. Stir in the cake flour. ...
7. Pour the batter into the pan. ...
8. Bake the cake for an hour and 15 minutes.

4 Ways to Bake a Cake - wikiHow
www.wikihow.com/**Bake-a-Cake**

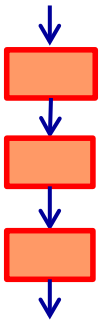
Control Structure: Sequence



Tips: How to read a C program?
You look for the "main()" and start reading it's "block"

Don't worry about the C syntax; we will discuss it next week. For now, just to show you how the algorithm is translated into the code. The logic remains the same, but you need to write the code according to the rules of the programming language.

Control Structures: **Sequence**



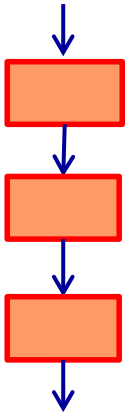
- Task: Compute the average of three integers
- How the program might look like

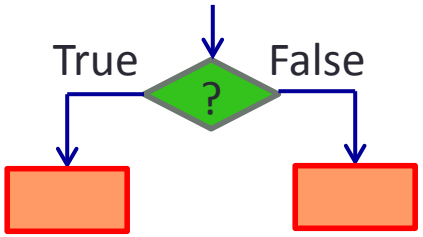
Unit3_prog1.c

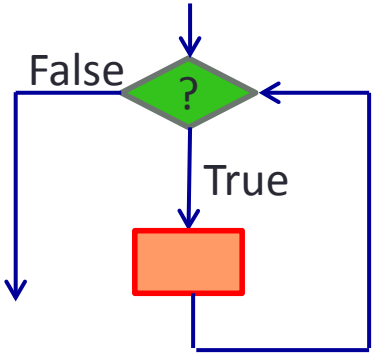
```
// This program computes the average of 3 integers
#include <stdio.h>


int main(void) {
   int num1, num2, num3;
   float ave;

   printf("Enter 3 integers: ");
   scanf("%d %d %d", &num1, &num2, &num3);

   ave = (num1 + num2 + num3) / 3.0;
   printf("Average = %.2f\n", ave);

   return 0;
}
```

Control Structures

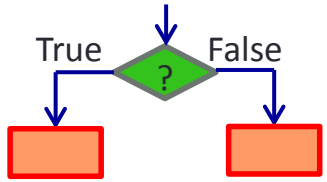
Sequence	• Default
Selection	• Also called branching
Repetition	• Also called loop





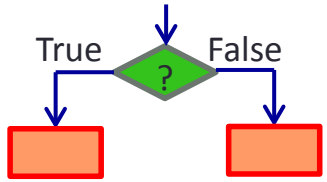


Control Structure: Selection



- If the player strikes a “3”
  - Give him a bear
- Else
  - Let him choose a water gun or a candy

Control Structure: Selection



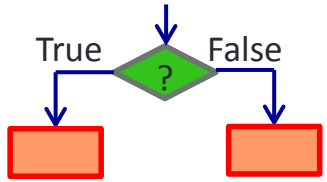

```
If (a condition is true)
  Do A
Else
  Do B
```

Can be MORE THAN one single instruction

- For example:

```
If (I have $1000000000000000)
  Buy a car
  Eat a lot of buffets
  Go travel
  Quit NUS!
Else
  Be good and study
```

Control Structure: Selection




```
If (a condition is true)
  Do A
Else
  Do B
```

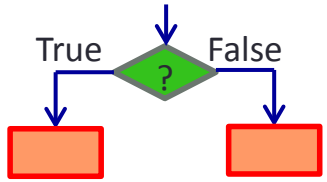
Can be MORE THAN one single instruction

- For example:

```
If (I have $1000000000000000)
  If (I am heartless)
    Buy a car
    Eat a lot of buffets
    Go travel
    Quit NUS!
  Else
       donate all the money to charity
Else
  Be good and study
```

Nested “if”

Control Structure: Selection



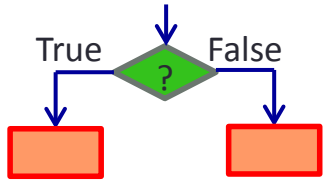

```
If (a condition is true)
  Do A
Else
  Do B
```

Can be **WITHOUT** "else"

- For example:

```
If (I have $1000000000000000)
  Buy a car
  Eat a lot of buffets
  Go travel
  Quit NUS!
Else
  Be good and study
```

Control Structures: **Selection (1/3)**



- Task: Arrange two integers in ascending order (sort)

Algorithm A:

enter values for *num1*, *num2*

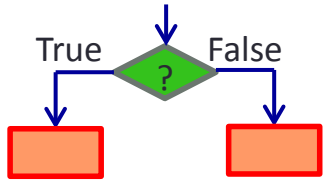
// Assign smaller number into final1,
// and larger number into final2
if (*num1* < *num2*)
  then *final1* ← *num1*
       final2 ← *num2*
  else *final1* ← *num2*
       final2 ← *num1*

// Transfer values in final1, final2 back to num1, num2
num1 ← *final1*
num2 ← *final2*

// Display sorted integers
print *num1*, *num2*

Variables used:

Control Structures: **Selection (2/3)**



- Task: Arrange two integers in ascending order (sort)

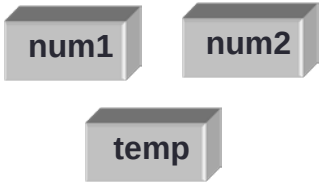
Algorithm B:

enter values for *num1*, *num2*

***//* Swap the values in the variables if necessary**
if (*num2* < *num1*)
  then *temp* ← *num1*
    num1 ← *num2*
    num2 ← *temp*

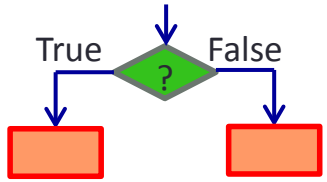
***//* Display sorted integers**
print *num1*, *num2*

Variables used:



Compare Algorithm A with Algorithm B.

Control Structures: **Selection (3/3)**



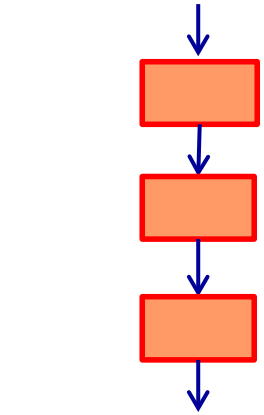
- How the program might look like for Algorithm B

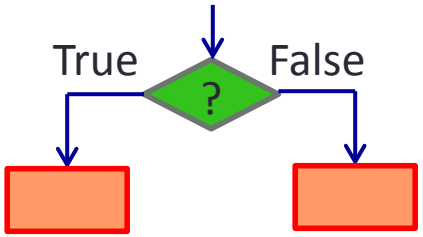
Unit3_prog2.c

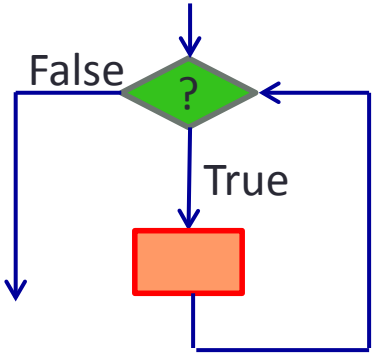
```
// This program arranges 2 integers in asce
#include <stdio.h>

int main(void) {
   int num1, num2, temp;

   printf("Enter 2 integers: ");
   scanf("%d %d", &num1, &num2);

   if (num2 < num1) {
      temp = num1;
      num1 = num2;
      num2 = temp;
   }
   printf("Sorted: num1 = %d, num2 = %d\n", num1, num2);
   return 0;
}
```

Control Structures

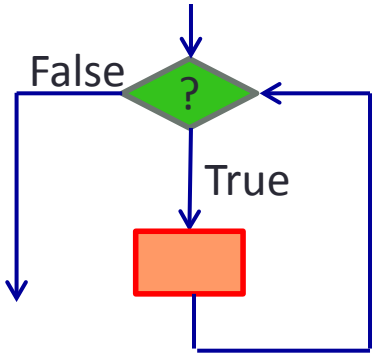
Sequence	• Default
Selection	• Also called branching
Repetition	• Also called loop





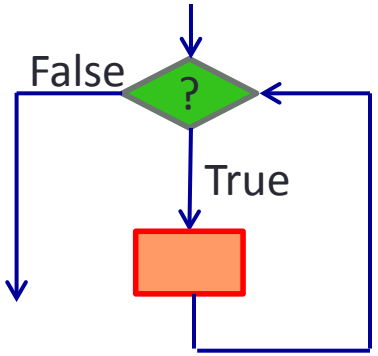


Control Structure: Repetition



- While there are prizes left
  - Play Plinko and give prizes

Control Structure: Repetition



- While (a condition)
  - Do something

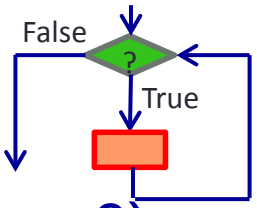
- For example

```
While (I am hungry)
  Eat a bun
```

- Again, can be more than one single instruction

```
While(I have money in bank)
  Take some money out from bank
  Eat an expensive meal
  While(I have money in my wallet)
    Go Shopping
```

Control Structures: **Repetition (1/3)**



- Task: Find sum of positive integers up to *n* (assume *n*>0)

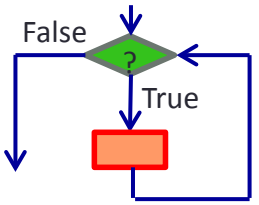
Algorithm:

```
enter value for n
// Initialise a counter count to 1, and ans to 0
count ← 1
ans ← 0
while (count ≤ n) do
    ans ← ans + count      // add count to ans
    count ← count + 1      // increase count by 1
// Display answer
print ans
```

Variables used:

Initialisation is very important!

Control Structures: **Repetition (2/3)**



- Important to trace pseudocode to check its correctness

Algorithm:

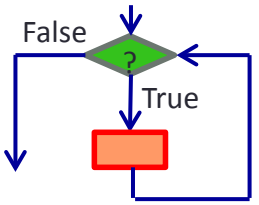
enter value for *n*
count ← 1
ans ← 0
while (*count* ≤ *n*) do
    ans ← *ans* + *count*
    count ← *count* + 1
  // Display answer
print *ans*

Assume user enters **3** for *n*.

(*count* <= *n*)?	*count*	*ans*
	1	0
true	2	1
true	3	3
true	4	6
false		

Output: **6**

Control Structures: **Repetition (3/3)**



- How the program might look like

Unit3_prog3.c

```
// Computes sum of positive integers up to n
#include <stdio.h>

int main(void) {
   int n; // upper limit
   int count = 1, ans = 0; // initialisation

   printf("Enter n: ");
   scanf("%d", &n);

   while (count <= n) {
      ans += count;
      count++;
   }
   printf("Sum = %d\n", ans);

   return 0;
}
```

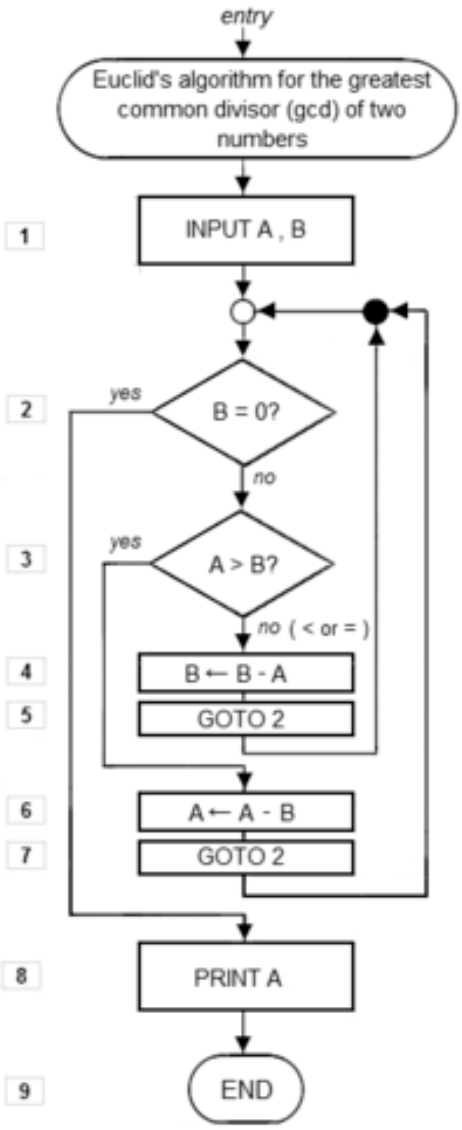
Euclid's Algorithm (1/3)

- To compute the greatest common divisor (GCD) of two integers
  - First documented algorithm by Greek mathematician Euclid in 300 B.C.
  - Also known as Euclidean Algorithm

1. Let A and B be integers with $A > B \geq 0$.
2. If $B = 0$, then the GCD is A and algorithm ends.
3. Otherwise, find q and r such that

   $$A = q \times B + r \quad \text{where } 0 \leq r < B$$

4. Replace A by B, and B by r. Go to step 2.



Euclid's Algorithm (2/3)

1. Let A and B be integers with $A > B \geq 0$.
2. If $B = 0$, then the GCD is A and algorithm ends.
3. Otherwise, find q and r such that

   $$A = q \times B + r \quad \text{where } 0 \leq r < B$$

4. Replace A by B, and B by r. Go to step 2.

- q is not important; r is the one that matters.
- r could be obtained by A modulo B (i.e. remainder of A / B)
- Assumption on $A > B$ unnecessary
- We will rewrite the algorithm

Euclid's Algorithm (3/3)

- Euclid's algorithm rewritten in modern form

```
// Assume A and B are non-negative
// integers, but not both zeroes.

Algorithm GCD(A, B) {
    while (B > 0) {
        r ← A modulo B
        A ← B
        B ← r
    }
    result is A
}
```

Let's trace GCD(12, 42)

(*B* > 0)?	*r*	*A*	*B*
		12	42
true	12	42	12
true	6	12	6
true	0	6	0
false			

Result: **6**

What is the difference between

Algorithm vs Program

- Algorithm
  - Ideas
  - Machine independent
- Program
  - The final code on a machine
  - Machine dependent

Research Idea vs Thesis

- Research Idea
  - Can be drawings, sketches, concepts, in any languages
- Thesis
  - Well-written documents in any languages, English, German, Chinese, etc…

Algorithm (1/3)

- An **algorithm** is a well-defined computational procedure consisting of *a set of instructions*, that takes some value or set of values as *input*, and produces some value or set of values as *output.*

'Algorithm' stems from 'Algoritmi', the Latin form of al-Khwārizmī, a Persian mathematician, astronomer and geographer.
Source: http://en.wikipedia.org/wiki/Algorithm

And computer will follow exactly

- "I asked my husband to peel half of the potatoes and put them on to boil…"
- And finally….

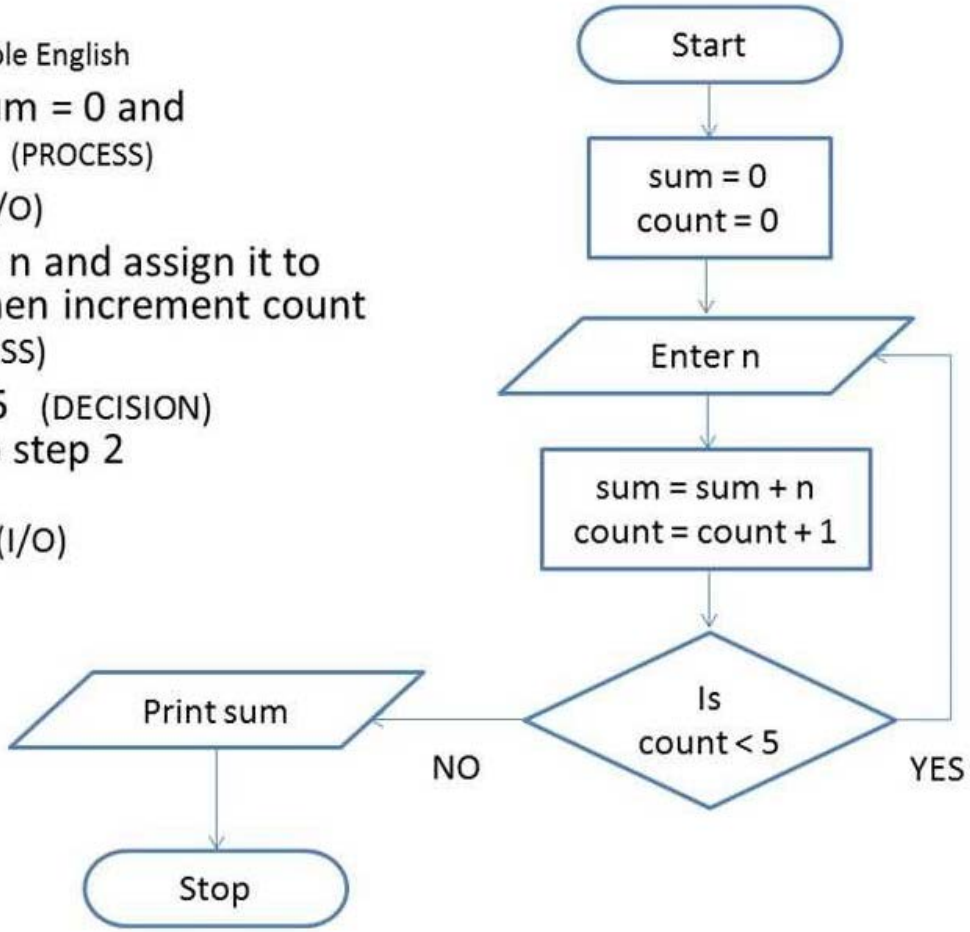
Algorithm: Example #1

Find the sum of 5 numbers

Algorithm in simple English

1. Initialize sum = 0 and count = 0 (PROCESS)
2. Enter n (I/O)
3. Find sum + n and assign it to sum and then increment count by 1 (PROCESS)
4. Is count < 5 (DECISION)
   if YES go to step 2
   else
   Print sum (I/O)

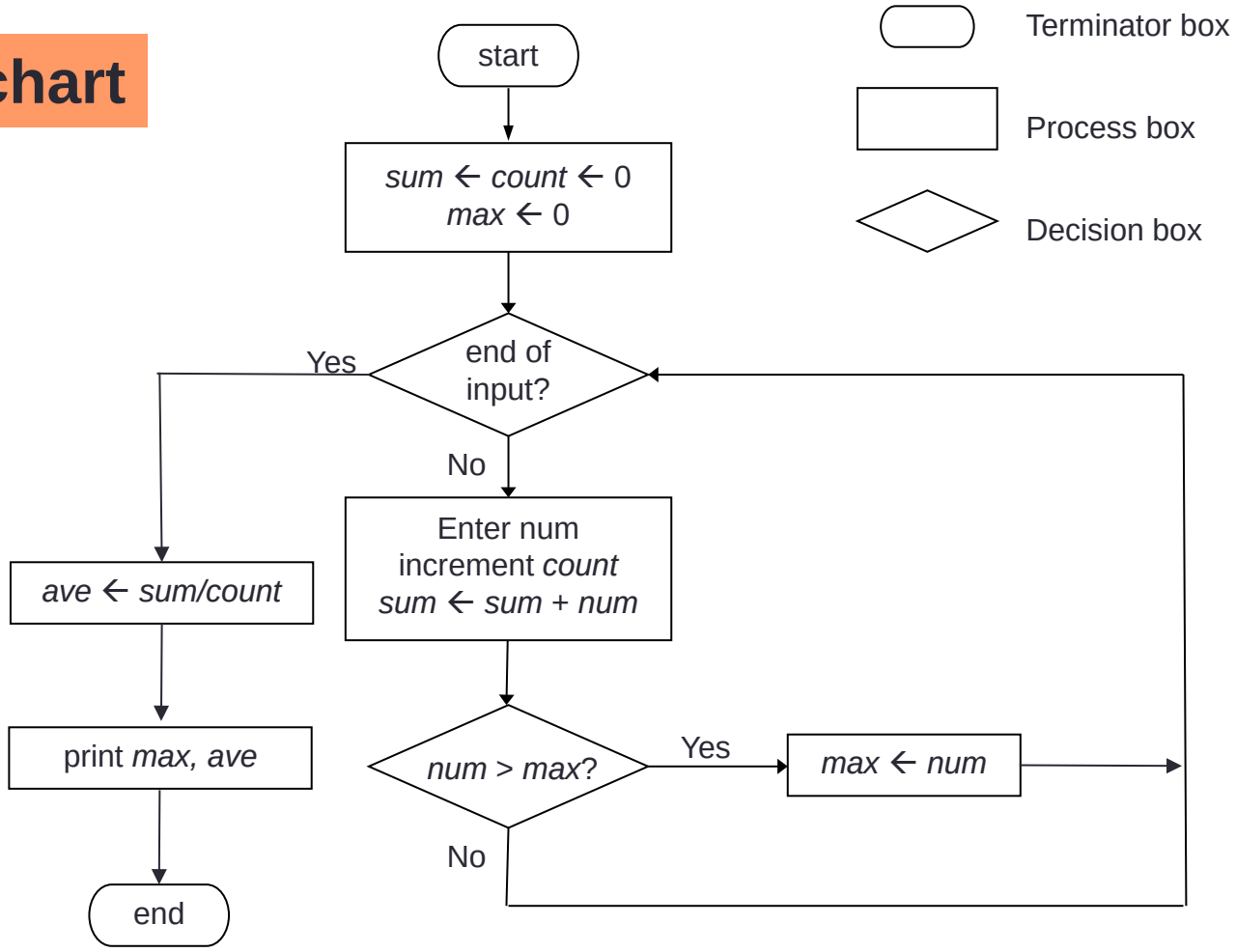
Flowchart



Algorithm: Example #2 (1/2)

- Find maximum and average of a list of numbers:

Flowchart



Algorithm: Example #2 (2/2)

- Find maximum and average of a list of numbers:

Pseudocode

The need to initialise variables.

```
sum ← count ← 0      // sum = sum of numbers
                     // count = how many numbers are entered?
max ← 0              // max to hold the largest value eventually
for each num entered,
    count ← count + 1
    sum ← sum + num
    if num > max
        then max ← num
ave ← sum / count

print max, ave
```

The need to indent.

Are there any errors in this algorithm?

Algorithm: Pseudocode

- We will write algorithms in pseudocode instead of flowchart as the former is more succinct
- However, there are no standard rules on how pseudocodes should look like
- General guidelines:
  - Every step must be unambiguous, so that anybody is able to hand trace the pseudocode and follow the logic flow
  - Use a combination of English (keep it succinct) and commonly understood notations (such as ← for assignment in our previous example)

Summary

- In this unit, you have learned about
  - The process of algorithmic problem solving
  - The properties of an algorithm
  - The three control structures
  - How to write algorithms in pseudocode
  - Tracing algorithms to verify their correctness